**Sentiment Analysis for Foody reviews**

Nguyễn Phương Hà

1. **Introduction**
   1. **Background**

Throughout the history of commerce, people have sought to validate the quality of products to inform their purchase decisions. This validation has become easier than ever in the age of widespread interconnectivity facilitated by the internet, with consumer reviews available for everything, from books (Goodreads), movies (IMDB), household goods on e-commerce sites (Amazon, Tiki, Shopee, etc.) to services like restaurants (Yelp, Foody) and accommodations (TripAdvisor). For businesses, good reviews can help to boost their customer base and increase their revenue, while bad reviews can significantly hurt their bottom line.

Restaurant owners and potential investors in food and beverage, specifically, have an interest in understanding the sentiment of customer reviews in order to make certain business decisions: for example, focusing on customer-favorite dishes while reducing supply of dishes that are not well-received, or providing more staff training if there is dissatisfaction with the current customer service.

* 1. **The project**

This project aims to build an NLP classification model to systematically predict whether a customer review is positive, neutral, or negative. I crawled available review data from Foody.vn, processed text data, built a self-training classifier to label the entire dataset based on a small amount of hand-labelled data, then built and optimized a classification model to classify all reviews into 1 of these 3 groups. Finally, I built a simple UI to allow users to input a review and receive a classification of that review.

* 1. **Relevant literature**

A key part of this project is label propagation using a semi-supervised learning. Semi-supervised learning is a widely-used machine learning methodology in cases where there is limited labelled data. A classifier is trained on the small amount of labelled data to make predictions on unlabelled samples, the most confident of which are added as “pseudo-labels” to re-train the classifier. This process is repeated until there are no more unlabelled data or a stop condition is reached [[1].](#References_4)

The scikit-learn library provides three semi-supervised learning estimators: Self-training classifier, Label Propagation, and Label Spreading. The Self-training classifier is the supervised classifier that can be iteratively re-trained to learn from unlabelled data. [[2]](#References_5) Meanwhile, Label Propagation and Label Spreading use a KNN-based approach to propagate labels based on similarities, with Label Spreading adding a regularization term so as to be more robust. [[3]](#References_5) In the current project, both a Self-training classifier and a Label Spreading model are tested against the benchmark to propagate labels to unlabelled samples using only a small set of hand-labelled data. Semi-supervised learning allowed this project to achieve high evaluation in the final classification model (overall accuracy = 94%) despite having initial labels for less than 10% of the dataset.

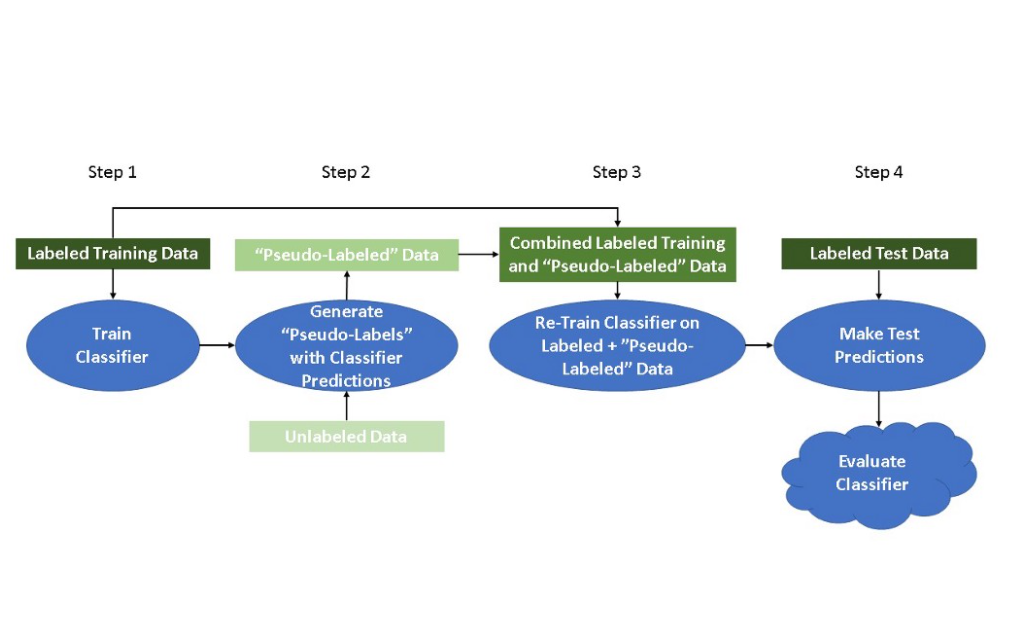


Figure 1. Self-training classification process. Source: Doug Steen, “A Gentle Introduction to Self-Training and Semi-Supervised Learning.” [[1]](#References_4)

The other model used in this study is a classification model to classify review text into one of three classes: positive, neutral, and negative. Five classification models were tested:

* **Logistic Regression**: classification using Sigmoid activation, where y = 1 if probability p >= 0.5 and y = 0 if p < 0.5
* **Decision Tree**: rule-based structure with each branch signifying a mutually exclusive rule based on input features
* **Random Forest Classifier**: an ensemble of many decision trees, final classification is based on majority vote
* **K-Nearest Neighbors**: classification based on distance to the nearest cluster of samples signifying a class
* **Support Vector Machine**: classification using support vectors to allow for more flexibility against outliers

1. **Data collection**

Using Selenium webdriver, I crawled a total of 5642 customer reviews of restaurants from 64 cities and provinces in Vietnam from the Foody review page. The collected data include (1) the full text of the review, and (2) the score given to the restaurant by the reviewer.

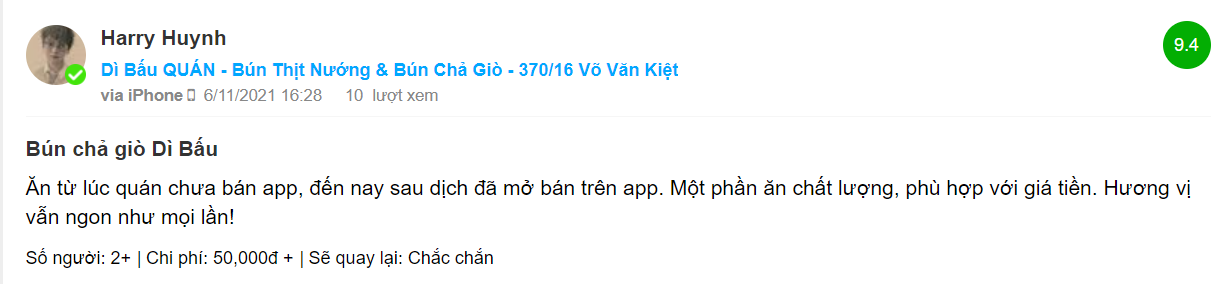


Figure 2. Sample customer review on Foody.vn. The collected data for this project include the score on the top left and the review text.

1. **EDA (Exploratory data analysis)**

After initially dropping duplicate rows (n = 78), which do not constitute a significant portion of the dataset and are more likely to be spam reviews and advertisements, EDA is performed on 5564 data points in total.

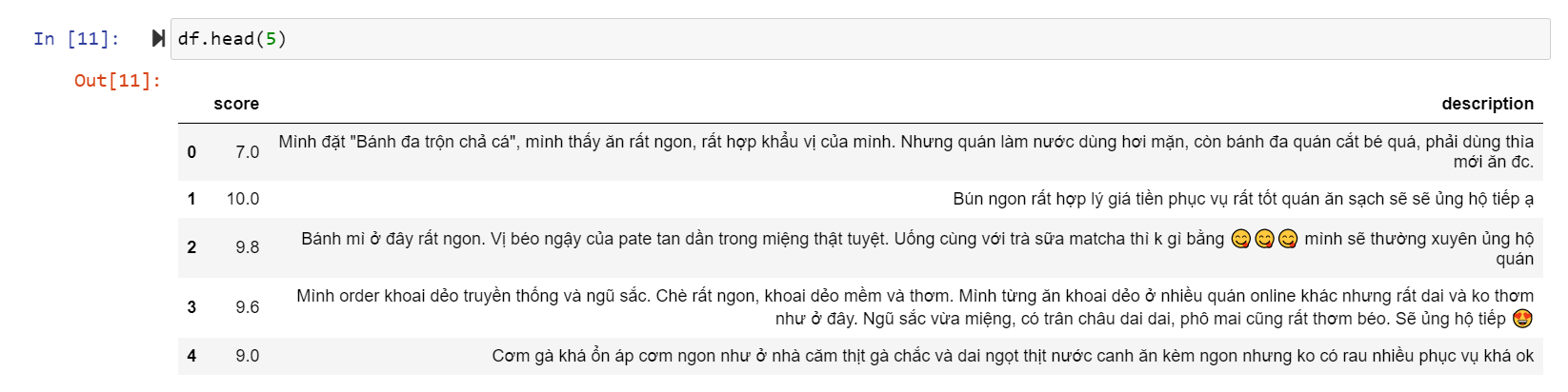
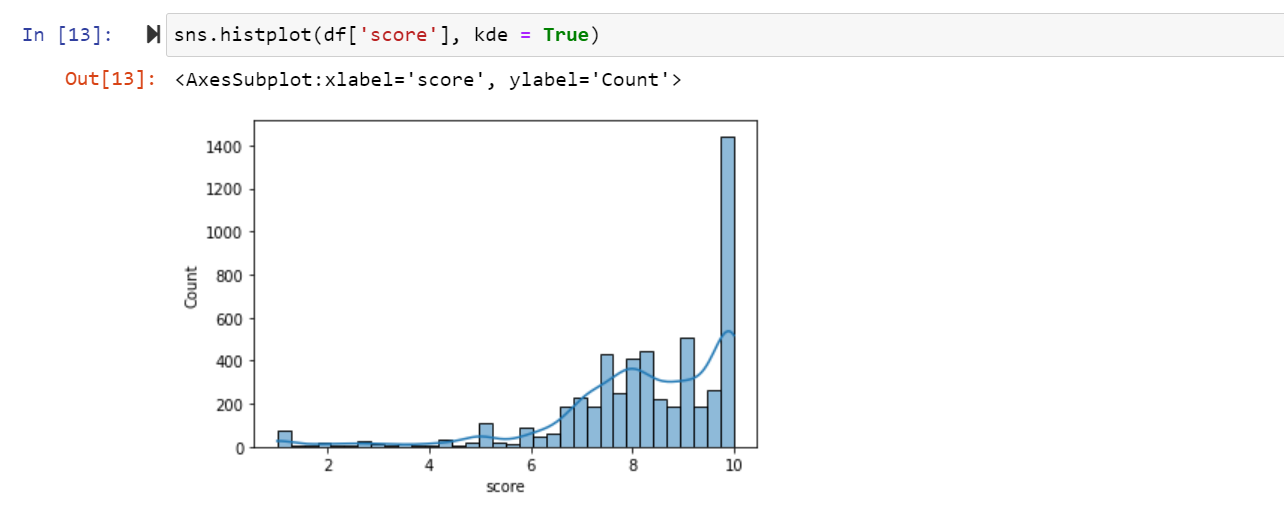
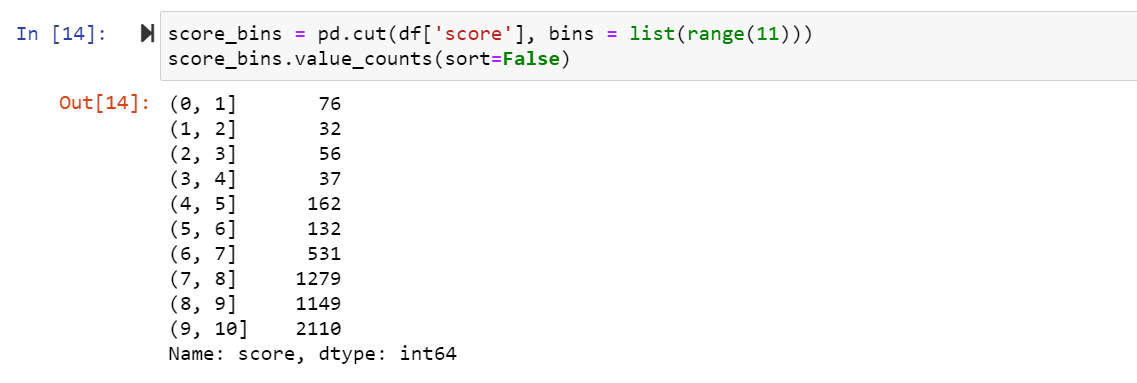


Figure 3. Sample of collected Foody customer reviews

* 1. **Review score**





Figures 4 and 5. Distribution of review scores from the dataset

The dataset is observed to be overly left-skewed: scores in the range of 8-10, which indicate a positive attitude, make up roughly 59% of the entire dataset, much more than neutral (scores 5-8) or negative scores (under 5). This imbalance indicate a need to re-sample the data at the feature engineering stage to avoid a bias in the classification model, most likely by over-sampling the data rather than under-sampling due to the small number of reviews with negative scores.

* 1. **Review text**
     1. **Cleaning and pre-processing text**

In addition to removing extra elements in the reviews that do not contain significant textual information (including numbers, punctuations, URLs, and hashtags) and changing all letters to lower-case, which are standard text processing steps in machine learning, I also used the Vietnamese positive/negative word processing procedure proposed in a similar sentiment analysis project by tuanpham1989. [[4]](#References_1) This procedure includes:

* Standardizing Vietnamese accent placement (for example, “òa” and “oà” are both standardized to “oà”)
* Standardizing common social media/texting spellings and abbreviations (for example, “ô kêi” and “okie” are standardized to “ok”)
* Replacing positive emojis with the word “tích\_cực” (“positive”) and negative emojis with the word “tiêu\_cực” (“negative”). The remaining emojis with no emotional connotation are removed.
* Replacing some words with their emotional connotations (for example, “hehe” is replaced with “tích\_cực”, “huhu” with “tiêu\_cực”)
* Replacing contradictions for some common positive/negative adjectives (for example, “không ngon” is replaced with “tiêu\_cực”)
* Adding features to common positive/negative adjectives (for example, if the review contains the word “ngon” without contradiction, then the word “tích\_cực” (“positive”) is added to the end of the review).

After pre-processing, the reviews are tokenized with the word\_tokenize module from Underthesea library. [[5]](#References_2)

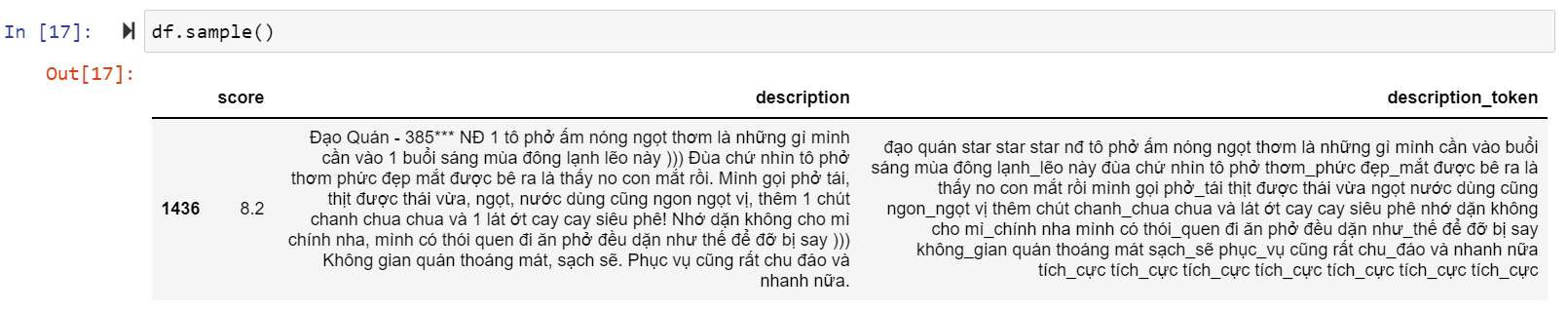


Figure 6. Sample review text after pre-processing and tokenizing

* + 1. **Issue with mislabelled data**

A closer observation of the group with “negative” scores reveal mislabelled data – some reviews that are highly positive or neutral nevertheless have very low scores. This mismatch indicates that review score is not a reliable indicator of the review text’s sentiment.

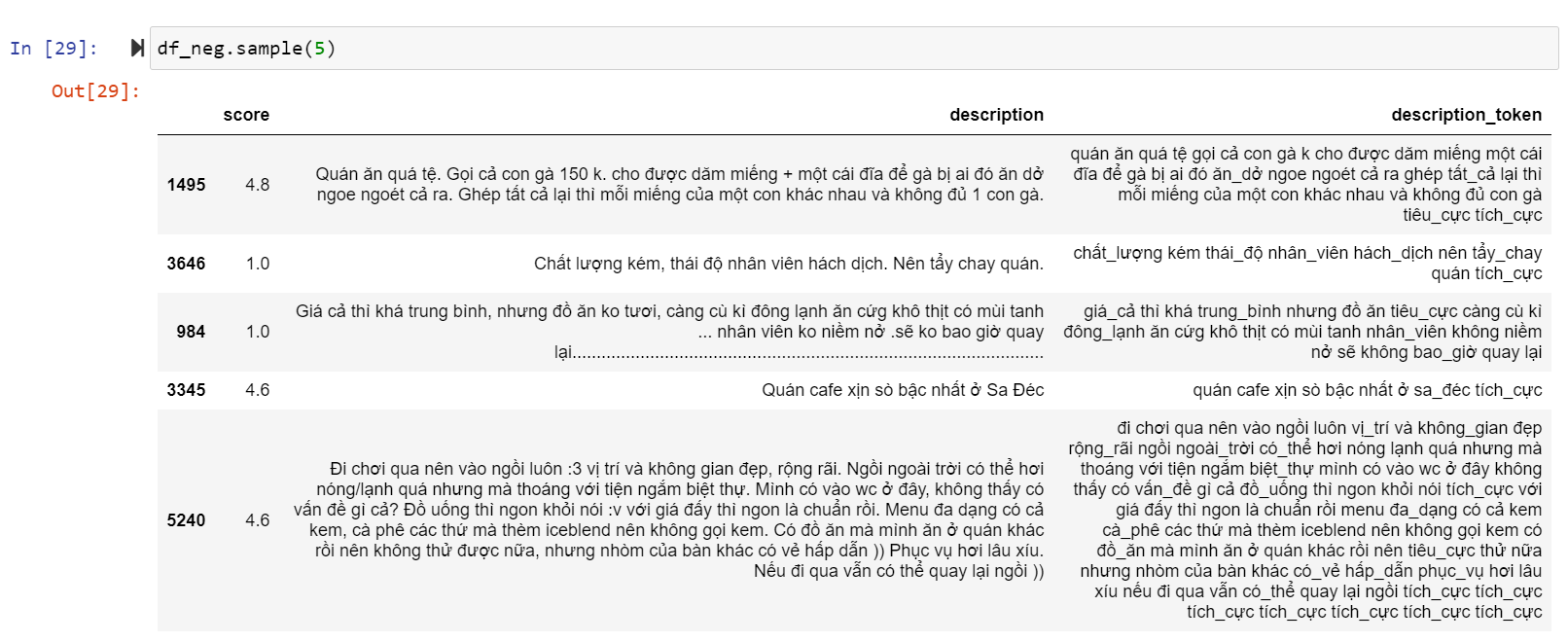


Figure 7. A sample of reviews with low scores. The fourth review in this sample

shows positive sentiment, while the fifth is neutral

1. **Re-labelling the dataset**
   1. **Re-labelling using semi-supervised learning**

Rather than use review score as a basis to classify the sentiment of each review, which has proven to be inaccurate, this project referred to semi-supervised learning to propagate labels based on a small subset of the dataset. [[6]](#References_3) First, I assigned the correct label (label = 0 for positive, 1 for neutral, and 2 for negative) to 508 samples by hand (9.3%), in the process dropping 89 more samples as irrelevant to the purpose of restaurant review sentiment analysis, leaving 4967 unlabelled samples (90.7%). The correct label set is further divided a training set and testing set, with n=300 for the training set (5.5% of the whole dataset) and n=208 for the testing set. The training set is balanced in terms of class samples (n=100 for each class) to prevent bias in the label propagation models.

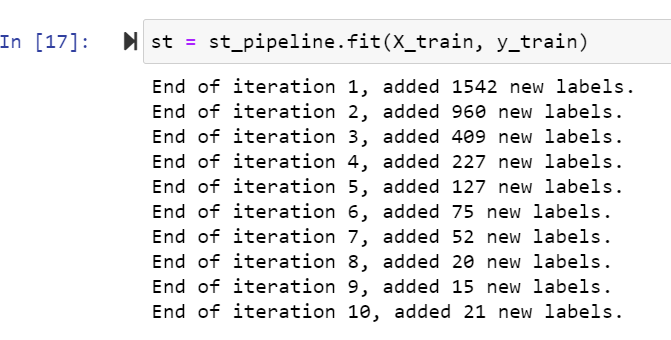
Consistent with [[6]](#References_3), I first trained and tested a Logistic Regression model with SGD optimization against hand-labelled data for benchmark. A semi-supervised model with the same base estimator is iteratively trained to add labels with at least 90% confidence on each iteration. Finally, a Label Spreading estimator with default parameters is tried.

The models are evaluated on micro-averaged F1 score to account for both precision and recall among classes.

Table 1. Semi-supervised method is benchmarked against supervised model

|  |  |
| --- | --- |
| **Method** | **Micro-averaged F1 score** |
| Supervised SGDClassifier | 0.69 |
| SelfTrainingClassifier | 0.83 |
| LabelSpreading | 0.28 |

As the self-training model has proven to be the most suitable for label propagation in this dataset, the model is fitted and used to predict all unlabelled data.



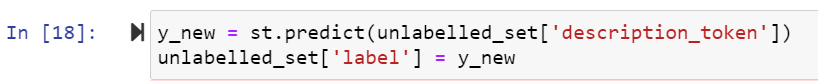


Figure 8. Labels with at least 90% confidence are added over each iteration

to train the classifier. Final fitted classifier is then used to

predict unlabelled dataset.

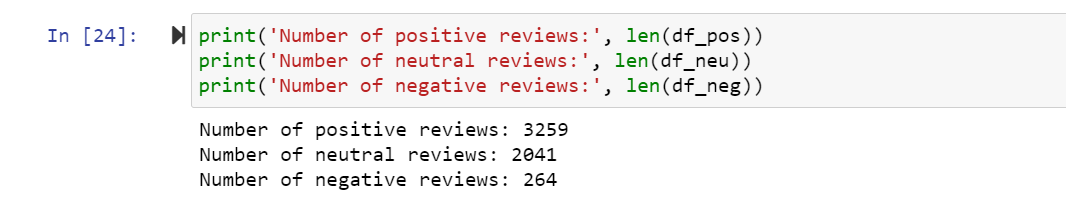


Figure 9a. Number of samples in each class before re-labelling

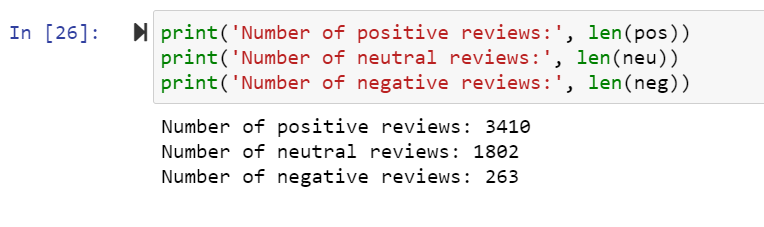


Figure 9b. Number of samples in each class after re-labelling

* 1. **Visualizing data after re-labelling**

Using TF-IDF, I extracted 300 words with highest TF-IDF scores that appear in fewer than 80% of the documents (max\_df = 0.8), along with their weighted frequencies. These words are visualized for all 3 groups: positive, neutral, and negative.



Figure 10a. Positive reviews

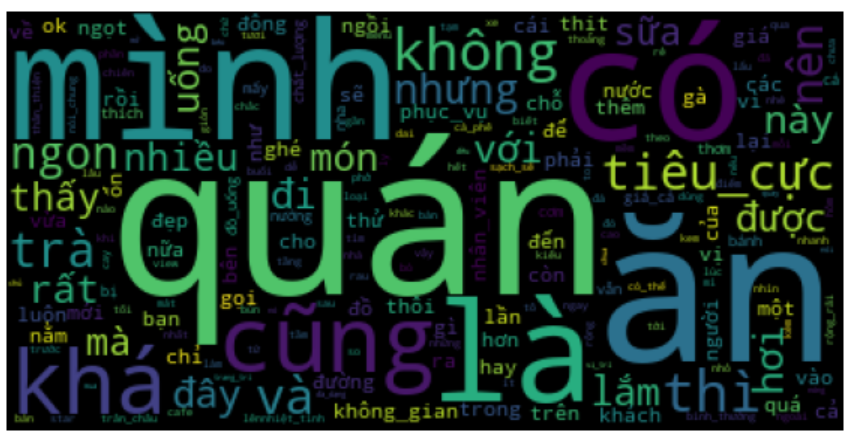


Figure 10b. Neutral reviews



Figure 10c. Negative reviews

From observation, the data seems consistent with expectations. The reviews with positive scores feature the word “ngon” (“delicious”) prominently, while neutral reviews start featuring words with more average to negative connotations like “cũng” (“too”), “khá” and “hơi” (“quite”, “a little”), and “tiêu\_cực” (“negative”).

Negative reviews are more likely to have “tiêu\_cực” (“negative”), although the word “tích\_cực” (“positive”) is still quite prevalent. This might be attributed to the remaining mislabelled data (the self-training label propagation model is only expected to be 83% accurate), as well as misleading or sarcastic expressions in the review text.

1. **Classification model**

After splitting the dataset into training (70%) and testing set (30%), I applied five classification models: Logistic Regression, Decision Tree, Random Forest, K-Nearest Neighbor, and Support Vector Machine with default parameters and under two conditions: without re-sampling to address the imbalanced dataset, and with oversampling using the SMOTE module from imblearn library. Under-sampling is not considered as a method for re-sampling, as the number of samples in the minority class (negative reviews) are too low.

Macro-averaged precision, recall, and F1 scores are used for evaluation to account for each model’s performance among the minority class.

Tables 2 and 3. Evaluation of five classification models with default parameters

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Without**  **re-sampling** | **Model** | **Accuracy** | **Precision** | **Recall** | **F1** |
| Logistic Regression | 0.92 | 0.93 | 0.75 | 0.80 |
| Decision Tree | 0.77 | 0.66 | 0.66 | 0.66 |
| Random Forest | 0.84 | 0.89 | 0.58 | 0.59 |
| KNN | 0.62 | 0.21 | 0.33 | 0.25 |
| SVC | 0.91 | 0.92 | 0.73 | 0.78 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **With**  **over-sampling** | **Model** | **Accuracy** | **Precision** | **Recall** | **F1** |
| Logistic Regression | 0.94 | 0.88 | 0.90 | 0.89 |
| Decision Tree | 0.77 | 0.63 | 0.67 | 0.65 |
| Random Forest | 0.85 | 0.87 | 0.70 | 0.75 |
| KNN | 0.62 | 0.60 | 0.36 | 0.31 |
| SVC | 0.92 | 0.92 | 0.75 | 0.80 |

Over-sampling is proven to be necessary to ensure the model is not biased towards the over-represented class (positive reviews). Logistic Regression is chosen for optimization as the highest and most consistent performer out of five classifiers.

The classifier is optimized using sklearn’s GridSearchCV function to choose the regularization term and algorithm that output the best micro-averaged F1 score – where precision and recall are balanced. The final optimized model achieves an 94% accuracy on the test set.

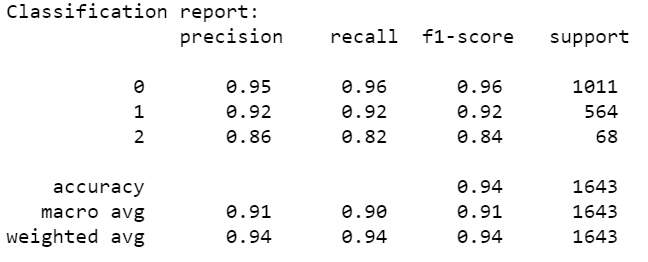


Figure 11. Classification report of the optimized classifier

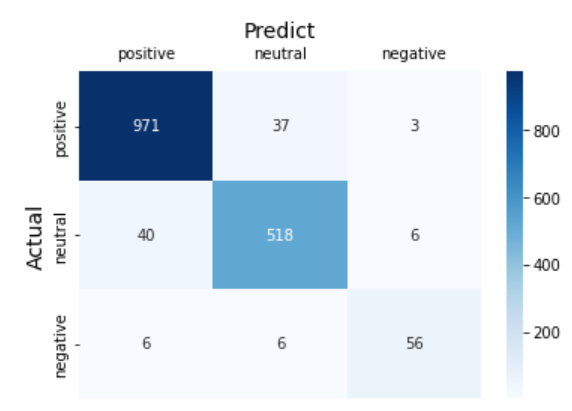


Figure 12. Confusion matrix

Understanding that the model was trained and tested on a dataset that included label-propagated data, I re-tested the model against hand-labelled data only. This outputs a 95% accuracy rate. I also used the model to re-predict the training set, receiving an accuracy score of 100% compared to 94% for the test set, suggesting that the model is slightly overfitted.

Finally, I built a simple user interface with Gradio that allows the user to input a review and receive a prediction on the sentiment of that review (Figure 13).



Figure 13. User interface to interact with the model

1. **Conclusion and future directions**

In this study, I built a classification model to predict whether a customer review of a restaurant is positive, neutral, or negative. After collecting and pre-processing review data from Foody.vn, I dropped review score as an unreliable operational definition for sentiment. I proceeded to hand-label 9.3% of total reviews, using scikit-learn’s SelfTrainingClassifier with base SGD-optimized Logistic Regression to propagate labels to the remaining 90.7% of the dataset, then trained and optimized a Logistic Regression model to predict this dataset with final accuracy score of 94%. I also built a simple UI to allow for model persistence and user interaction.

The limited availability of correctly labelled data in this study (n=508 hand-labelled reviews out of 5475 total samples) undoubtedly had an impact on the final model’s accuracy and ability to generalize to unseen data, even with high-confidence label propagation. With less time constraint, performance might be boosted by ensuring total label accuracy for the entire dataset before training the classification model.

**References**

[1] Doug Steen, “A Gentle Introduction to Self-Training and Semi-Supervised Learning.” <https://towardsdatascience.com/a-gentle-introduction-to-self-training-and-semi-supervised-learning-ceee73178b38>

[2] Scikit-learn Documentation, 1.14. Semi-supervised Learning. <https://scikit-learn.org/stable/modules/semi_supervised.html>

[3] Gaëlle Guillou, “Finding a Balance With Semi-Supervised Learning.” <https://blog.dataiku.com/finding-a-balance-with-semi-supervised-learning>

[4] tuanpham1989 sentiment\_analysis\_nal Github repository: <https://github.com/tuanpham1989/sentiment_analysis_nal>

[5] Underthesea – Vietnamese NLP toolkit: <https://pypi.org/project/underthesea/>

[6] Scikit-learn auto-example “Semi-supervised Classification on a Text Dataset”: <https://scikit-learn.org/0.24/auto_examples/semi_supervised/plot_semi_supervised_newsgroups.html#sphx-glr-download-auto-examples-semi-supervised-plot-semi-supervised-newsgroups-py>